1.create multilevel inheritance for

//Vehicle
//Four_wheeler
//Petrol_Four_Wheeler
//FiveSeater_Petrol_Four_Wheeler
//Baleno_FiveSeater_Petrol_Four_Wheeler

```
package Assesement_day4;
class Vehicle {
void displayVehicle() {
System.out.println(" Vehicle");
}
}

class Four_wheeler extends Vehicle {
void displayFourWheeler() {
System.out.println(" Four Wheeler");
}
}

class Petrol_Four_Wheeler extends Four_wheeler {
void displayPetrolFourWheeler() {
System.out.println(" Petrol Four Wheeler");
}
}

class FiveSeater_Petrol_Four_Wheeler extends
Petrol_Four_Wheeler {
void displayFiveSeaterPetrolFourWheeler() {
```

```
System.out.println("Five Seater Petrol Four Wheeler");
}
}

class Baleno_FiveSeater_Petrol_Four_Wheeler extends
FiveSeater_Petrol_Four_Wheeler {
void displayBaleno() {
System.out.println(" Baleno Five Seater Petrol Four Wheeler");
}

public static void main(String[] args) {
Baleno_FiveSeater_Petrol_Four_Wheeler baleno = new
Baleno_FiveSeater_Petrol_Four_Wheeler();
baleno.displayVehicle();
baleno.displayFourWheeler();
baleno.displayPetrolFourWheeler();
baleno.displayFiveSeaterPetrolFourWheeler();
baleno.displayBaleno();
}
}
```

Output:

```
  Vehicle
Four Wheeler
Petrol Four Wheeler
Five Seater Petrol Four Wheeler
  Baleno Five Seater Petrol Four Wheeler
```

2. Demonstrate the use of the super keyword

```
class Vehicle {
	String st = " Vehicle";
	void display() {
		System.out.println("This is a Vehicle");
	}
}

class Car extends Vehicle {
	String st = "Car Brand";
	void display() {
		System.out.println("This is a Car");
	}
	void show() {
		System.out.println("Child brand: " + st);
		System.out.println("Parent brand: " + super.st);
		display();
		super.display();
	}
}

public class Main {
	public static void main(String[] args) {
		Car car = new Car();
		car.show();
	}
}
```

Output:

Child st: Car Brand
Parent st: Generic Vehicle
This is a Car
This is a Vehicle

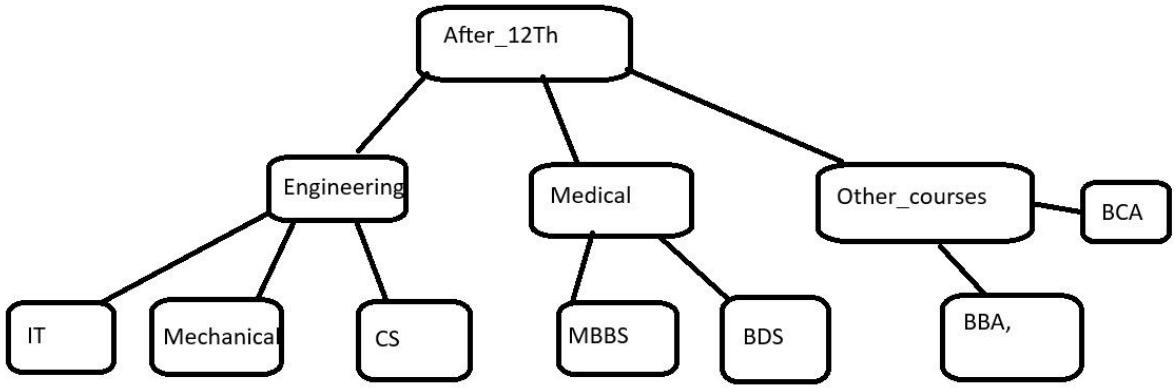
3. Create Hospital super class and access this class inside the patient child class and access properties from Hospital class.

```
package Assesement_day4;
class Hospital {
String name;

Hospital(String name) {
this.name = name;
}
}

class Patient extends Hospital {
String patientName;

Patient(String hospitalName, String patientName) {
super(hospitalName);
this.patientName = patientName;
}

void displayInfo() {
System.out.println("Hospital Name: " + name);
System.out.println("Patient Name: " + patientName);
}
}
```

```
public class Hospital_patient {

public static void main(String[] args) {
Patient patient = new Patient("Apollo Hospital", "ABC");
patient.displayInfo();

}

}
```

Output:
Hospital Name: Apollo Hospital
Patient Name: ABC

4. Create Hierarchical inheritance




```
package Assesement_day4;
// Parent class
class After12th {
	void displayOptions() {
		System.out.println("Options after 12th: Engineering,
Medical, Other Courses");
	}
```

```
}

class Engineering extends After12th {
	void showBranches() {
		System.out.println("Engineering Branches: IT, 
Mechanical, CS");
	}
}

class Medical extends After12th {
	void showBranches() {
		System.out.println("Medical Courses: MBBS, BDS");
	}
}

class OtherCourses extends After12th {
	void showBranches() {
		System.out.println("Other Courses: BBA, BCA");
	}
}

class IT extends Engineering {
	void courseDetails() {
		System.out.println("IT: Information Technology 
focuses on software, networking, and data.");
	}
}

class Mechanical extends Engineering {
	void courseDetails() {
```

```
            System.out.println("Mechanical: Focus on
machinery, design, and manufacturing.");
        }
}

class CS extends Engineering {
        void courseDetails() {
            System.out.println("CS: Computer Science focuses
on programming, AI, and algorithms.");
        }
}

class MBBS extends Medical {
        void courseDetails() {
            System.out.println("MBBS: Bachelor of Medicine,
Bachelor of Surgery.");
        }
}

class BDS extends Medical {
        void courseDetails() {
            System.out.println("BDS: Bachelor of Dental
Surgery.");
        }
}

class BBA extends OtherCourses {
        void courseDetails() {
            System.out.println("BBA: Bachelor of Business
Administration.");
        }
```

```
}

class BCA extends OtherCourses {
	void courseDetails() {
		System.out.println("BCA: Bachelor of Computer
Applications.");
	}
}

public class Hierarchical{
	public static void main(String[] args) {
		// Parent object
		After12th after12th = new After12th();
		after12th.displayOptions();
		System.out.println();


		Engineering eng = new Engineering();
		eng.displayOptions();
		eng.showBranches();
		IT it = new IT();
		it.displayOptions();
		it.showBranches();
		it.courseDetails();
		Mechanical mech = new Mechanical();
		mech.courseDetails();
		CS cs = new CS();
		cs.courseDetails();
		System.out.println();

		Medical med = new Medical();
```

```
            med.displayOptions();
            med.showBranches();
            MBBS mbbs = new MBBS();
            mbbs.courseDetails();
            BDS bds = new BDS();
            bds.courseDetails();
            System.out.println();

            OtherCourses oc = new OtherCourses();
            oc.displayOptions();
            oc.showBranches();
            BBA bba = new BBA();
            bba.courseDetails();
            BCA bca = new BCA();
            bca.courseDetails();
        }
}
```

Output:

```
Options after 12th: Engineering, Medical, Other Courses

Options after 12th: Engineering, Medical, Other Courses
Engineering Branches: IT, Mechanical, CS
Options after 12th: Engineering, Medical, Other Courses
Engineering Branches: IT, Mechanical, CS
IT: Information Technology focuses on software, networking, and data.
Mechanical: Focus on machinery, design, and manufacturing.
CS: Computer Science focuses on programming, AI, and algorithms.

Options after 12th: Engineering, Medical, Other Courses
```

Medical Courses: MBBS, BDS
MBBS: Bachelor of Medicine, Bachelor of Surgery.
BDS: Bachelor of Dental Surgery.

Options after 12th: Engineering, Medical, Other Courses
Other Courses: BBA, BCA
BBA: Bachelor of Business Administration.
BCA: Bachelor of Computer Applications.

Polymorphism

1. Create a class Calculator with the following overloaded add()
    1. add(int a, int b)
    2. add(int a, int b, int c)
    3. add(double a, double b)

```
package Assesement_day4;
public class Calculator {
   public int add(int a, int b)
    {
            return a + b;
      }

   public int add(int a, int b, int c)
    {
            return a + b + c;
      }

   public double add(double a, double )

    {
```

```
        return a + b;
}
    public static void main(String[] args) {
        Calculator calculator = new Calculator();
        System.out.println("Adding two integers: " +
calculator.add(5, 7));
        System.out.println("Adding three integers: " +
calculator.add(5, 7, 9));
        System.out.println("Adding two doubles: " +
calculator.add(5.5, 7.7));
    }
}
```

Output:

```
Adding two integers: 12
Adding three integers: 21
Adding two doubles: 13.2
```

2. Create a base class Shape with a method area() that prints a message.
   Then create two subclasses
   Circle→override area() to calculator and print area of circle
   Rectangle→ override area() to calculate and print area of a rectangle

```
package Assesement_day4;
```

```
class Shape {
	void area() {
		System.out.println("Calculating area of shape");
	}
}

class Circle extends Shape {
	double radius;

	Circle(double radius) {
		this.radius = radius;
	}

	void area() {
		double area = Math.PI * radius * radius;
		System.out.println("Area of circle: " + area);
	}
}

class Rectangle extends Shape {
	double length;
	double width;

	Rectangle(double length, double width) {
		this.length = length;
		this.width = width;
	}

	void area() {
		double area = length * width;
		System.out.println("Area of rectangle: " + area);
```

```
    }
}
public class circle_rectangle {
    public static void main(String[] args) {
        Shape circle = new Circle(5.0);
        circle.area();

        Shape rectangle = new Rectangle(4.0, 6.0);
        rectangle.area();
    }
}
```

Output:

```
Area of circle: 78.53981633974483
Area of rectangle: 24.0
```

3. Create a Bank class with a method

getInterestRate()

create subclasses:

SBI→return 6.7%

ICICI→return 7.0%

HDFC→return 7.5%

```
package Assesement_day4;
abstract class Bank {
    abstract double getInterestRate();
}

class SBI extends Bank {
    double getInterestRate() {
```

```
        return 6.7;
    }
}

class ICICI extends Bank {
    double getInterestRate() {
        return 7.0;
    }
}

class HDFC extends Bank {
    double getInterestRate() {
        return 7.5;
    }
}

public class Bank_class {
    public static void main(String[] args) {
        Bank[] banks = {new SBI(), new ICICI(), new HDFC()};

        for (Bank bank : banks) {

System.out.println(bank.getClass().getSimpleName() + "
Interest Rate: " + bank.getInterestRate() + "%");
        }
    }
}
```

Output:

SBI Interest Rate: 6.7%

ICICI Interest Rate: 7.0%
HDFC Interest Rate: 7.5%

2.Design an interface Bank with methods deposit(), withdraw(), and getBalance(). Implement this in SavingsAccount and CurrentAccount classes.

- Use inheritance to create a base Account class.
- Demonstrate method overriding with customized logic for withdrawal (e.g., minimum balance in SavingsAccount).

```
package Assesement_day4;
    interface Bank {
        void deposit(double amount);
        void withdraw(double amount);
        double getBalance();
}

abstract class Account implements Bank {
        double balance;

        Account(double balance) {
                this.balance = balance;
        }

        public void deposit(double amount) {
                balance += amount;
                System.out.println("Deposited: " + amount);
```

```
    }

    public double getBalance() {
        return balance;
    }
}

class SavingsAccount extends Account {
    SavingsAccount(double balance) {
        super(balance);
    }

    public void withdraw(double amount) {
        if (balance - amount >= 1000) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Insufficient balance");
        }
    }
}

class CurrentAccount extends Account {
    CurrentAccount(double balance) {
        super(balance);
    }

    public void withdraw(double amount) {
        balance -= amount;
        System.out.println("Withdrawn: " + amount);
    }
```

```
}

public class Bank{
	public static void main(String[] args) {
		Bank savings = new SavingsAccount(1500);
		savings.deposit(500);
		savings.withdraw(1000);
		System.out.println("Savings Balance: " +
savings.getBalance());

		Bank current = new CurrentAccount(1000);
		current.deposit(500);
		current.withdraw(1500);
		System.out.println("Current Balance: " +
current.getBalance());
	}
}
```

Output:

```
Deposited: 500.0
Withdrawn: 1000.0
Savings Balance: 1000.0
Deposited: 500.0
Withdrawn: 1500.0
Current Balance: 0.0
```

3.Create a base class Vehicle with method start(). Derive Car, Bike, and Truck from it and override the start() method.

- Create a static method that accepts Vehicle type and calls start().
- Pass different vehicle objects to test polymorphism.

```
package Assesement_day4;
	class Vehicle {
		void start() {
			System.out.println("Vehicle started");
		}
	}

	class Car extends Vehicle {
		void start() {
			System.out.println("Car started");
		}
	}

	class Bike extends Vehicle {
		void start() {
			System.out.println("Bike started");
		}
	}

	class Truck extends Vehicle {
		void start() {
			System.out.println("Truck started");
		}
	}
```

```
public class Vehicle {
	static void startVehicle(Vehicle vehicle) {
		vehicle.start();
	}

	public static void main(String[] args) {
		Vehicle car = new Car();
		Vehicle bike = new Bike();
		Vehicle truck = new Truck();

		startVehicle(car);
		startVehicle(bike);
		startVehicle(truck);
	}
}
```

Output:

```
Car started
Bike started
Truck started
```

4.Design an abstract class Person with fields like name, age, and abstract method getRoleInfo().

Create subclasses:

- Student: has course and roll number.

- Professor: has subject and salary.
- TeachingAssistant: extends Student and implements getRoleInfo() in a hybrid way.
- Create and print info for all roles using overridden getRoleInfo().

```
package Assesement_day4;
abstract class Person {
	String name;
	int age;

	Person(String name, int age) {
		this.name = name;
		this.age = age;
	}

	abstract void getRoleInfo();
}

class Student extends Person {
	String course;

	Student(String name, int age, String course) {
		super(name, age);
		this.course = course;
	}

	void getRoleInfo() {
```

```
            System.out.println("Name: " + name + ", Age: "
+ age + ", Course: " + course + ", Role: Student");
        }
}

class Professor extends Person {
        String subject;

        Professor(String name, int age, String subject) {
            super(name, age);
            this.subject = subject;
        }

        void getRoleInfo() {
            System.out.println("Name: " + name + ", Age: "
+ age + ", Subject: " + subject + ", Role: Professor");
        }
}

class TeachingAssistant extends Student {
        TeachingAssistant(String name, int age, String
course) {
            super(name, age, course);
        }

        void getRoleInfo() {
            super.getRoleInfo();
            System.out.println("Additional Role: Teaching
Assistant");
```

```
        }
}

public class person_with_different_fields{
        public static void main(String[] args) {
                Person student = new Student("Sanjana", 20,
"B.Tech");
                Person professor = new Professor("Dr. Dhana",
40, "Mathematics");
                Person teachingAssistant = new
TeachingAssistant("Sri", 22, "B.Tech");

                student.getRoleInfo();
                professor.getRoleInfo();
                teachingAssistant.getRoleInfo();
        }
}
```

Output:

```
Name: Sanjana, Age: 20, Course: B.Tech, Role: Student
Name: Dr. Dhana, Age: 40, Subject: Mathematics,
Role: Professor
Name: Sri, Age: 22, Course: B.Tech, Role: Student
Additional Role: Teaching Assistant
```

5.Create:

- Interface Drawable with method draw()
- Abstract class Shape with abstract method area()
  Subclasses: Circle, Rectangle, and Triangle.
- Calculate area using appropriate formulas.
- Demonstrate how interface and abstract class work together.

```
package Assesement_day4;
    interface Drawable {
        void draw();
    }

    abstract class Shape implements Drawable {
        abstract double area();
    }

    class Circle extends Shape {
        double radius;

        Circle(double radius) {
            this.radius = radius;
        }

        public void draw() {
            System.out.println("Drawing circle");
        }
```

```
    double area() {
        return 3.14 * radius * radius;
    }
}

class Rectangle extends Shape {
    double length;
    double width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    public void draw() {
        System.out.println("Drawing rectangle");
    }

    double area() {
        return length * width;
    }
}

class Triangle extends Shape {
    double base;
    double height;

    Triangle(double base, double height) {
        this.base = base;
```

```
        this.height = height;
    }

    public void draw() {
        System.out.println("Drawing triangle");
    }

    double area() {
        return 0.5 * base * height;
    }
}

public class Drawable_shapes {
    public static void main(String[] args) {
        Shape circle = new Circle(5);
        Shape rectangle = new Rectangle(4, 6);
        Shape triangle = new Triangle(3, 7);

        circle.draw();
        System.out.println("Circle area: " +
circle.area());

        rectangle.draw();
        System.out.println("Rectangle area: " +
rectangle.area());

        triangle.draw();
        System.out.println("Triangle area: " +
triangle.area());
```

```
        }
}
```

Output:

```
Drawing circle
Circle area: 78.5
Drawing rectangle
Rectangle area: 24.0
Drawing triangle
Triangle area: 10.5
```